(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
31 décembre 2003 (31.12.2003)

PCT

(10) Numéro de publication internationale
WO 2004/002058 A2

(51) Classification internationale des brevets[7] : H04L 9/30

(21) Numéro de la demande internationale :
PCT/FR2003/001871

(22) Date de dépôt international : 18 juin 2003 (18.06.2003)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
02/07688 19 juin 2002 (19.06.2002) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **GEMPLUS** [FR/FR]; Parc d'Activités de Gémenos, Avenue du Pic-de-Bertagne, F-13420 Gémenos (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **FEYT, Nathalie** [FR/FR]; 8, chemin de Raphèle, 7 lotissement l'Oliveraie, F-13780 Cuges les Pins (FR). **JOYE, Marc** [FR/FR]; 19, rue Voltaire, F-83640 Saint Zacharie (FR).

(74) Mandataire : **AIVAZIAN, Denis**; Gemplus la Vigie, Service brevets, BP 100, F-13705 La Ciotat Cedex (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) États désignés *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Déclarations en vertu de la règle 4.17 :

— *relative à l'identité de l'inventeur (règle 4.17.i)) pour les désignations suivantes AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

— *relative au droit du déposant de demander et d'obtenir un brevet (règle 4.17.ii)) pour les désignations suivantes AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

[Suite sur la page suivante]

(54) **Title:** METHOD OF GENERATING ELECTRONIC KEYS FOR A PUBLIC-KEY CRYPTOGRAPHY METHOD AND A SECURE PORTABLE OBJECT USING SAID METHOD

(54) **Titre :** PROCEDE DE GENERATION DE CLES ELECTRONIQUES POUR PROCEDE DE CRYPTOGRAPHIE A CLE PUBLIQUE ET OBJET PORTATIF SECURISE METTANT EN OEUVRE LE PROCEDE

(57) **Abstract:** The invention relates to a method of generating electronic keys (d) for a public-key cryptography method using an electronic device. The inventive method comprises two separate calculation steps, namely: step A consisting in (i) calculating pairs of prime numbers (p, q), said calculation being independent of knowledge of the pair (e, l) in which e is the public exponent and l is the length of the key of the cryptography method, and (ii) storing the pairs thus obtained; and step B which is very quick and can be executed in real time by the device, consisting in calculating a key d from the results of step A and knowledge of the pair (e, l).

(57) **Abrégé :** L'invention concerne un procédé de génération de clés électroniques d pour procédé de cryptographie à clé publique au moyen d'un dispositif électronique. Selon l'invention, le procédé comprend deux étapes de calcul dissociées. Une étape A consiste à - calculer des couples de nombres premiers (p, q), ce calcul est indépendant de la connaissance du couple (e, l) e l'exposant public et l la longueur de la clé du procédé de cryptographie et à - stocker les couples ainsi obtenus. Une étape B très rapide qui peut être exécutée en temps réel par le dispositif, consiste à calculer une clé d à partir des résultats de l'étape A et de la connaissance du couple (e, l).

WO 2004/002058 A2

— *relative au droit du déposant de revendiquer la priorité de la demande antérieure (règle 4.17.iii)) pour toutes les désignations*
— *relative à la qualité d'inventeur (règle 4.17.iv)) pour US seulement*

Publiée :
— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

PROCEDE DE GENERATION DE CLES ELECTRONIQUES POUR PROCEDE DE CRYPTOGRAPHIE A CLE PUBLIQUE ET OBJET PORTATIF SECURISE METTANT EN ŒUVRE LE PROCEDE

L'invention concerne un procédé de génération de clés électroniques pour procédé de cryptographie à clé publique. Elle concerne également un objet portatif sécurisé mettant en œuvre le procédé.

L'invention concerne plus particulièrement la génération de clés d'un système cryptographique de type RSA et leur stockage sur un objet sécurisé en vue de leur utilisation dans une application nécessitant de la sécurité.

L'invention s'applique tout particulièrement à dés objets sécurisés ne possédant pas d'importante ressource mémoire telle que de la mémoire électriquement programmable, ni de ressources de calcul puissantes comme c'est le cas pour les cartes à puce.

Une application de l'invention est le commerce électronique par l'intermédiaire d'un téléphone portable. Dans ce contexte les clés peuvent se trouver sur la carte SIM du téléphone.

Il est en effet prévu que certains programmes d'applications utilisent de telles clés pour mettre en œuvre un transfert de données confidentielles, dans un contexte de commerce électronique par exemple. Par la suite, on considèrera que ces applications sont fournies par une entité fournisseur de service.

En outre, il est connu que pour garantir l'intégrité de la clé, on lui associe généralement un certificat fourni par une entité de confiance.

Parmi les procédés de cryptographie à clé publique, on s'intéresse dans ce qui suit au protocole de cryptographie RSA (Rivest Shamir et Adleman). Ce protocole met en œuvre une étape de génération de nombres premiers de grande taille, coûteuse en temps de calcul et en place mémoire.

On rappelle que ce protocole de cryptographie RSA permet le chiffrement d'informations et/ou l'authentification entre deux entités et/ou la signature électronique de messages.

Le protocole de cryptographie RSA est le plus utilisé car il possède des propriétés qui lui permettent d'être employé aussi bien en chiffrement qu'en génération de signature.

Pour ce faire, le système de cryptographie RSA comprend un algorithme « public » réalisant la fonction de chiffrement ou de vérification de signature et un algorithme « privé » réalisant la fonction de déchiffrement ou de génération de signature.

Sa sécurité repose sur la difficulté de factorisation d'un nombre entier public N de grande taille qui est le produit de deux nombres premiers secrets p et q de grande taille, le couple (p,q) entrant dans le calcul de la clé secrète d utilisée par la fonction de déchiffrement ou par la fonction de calcul d'une signature.

Afin de mieux comprendre le problème qui va être exposé dans la suite, on va rappeler dans ce qui suit les paramètres entrant dans un schéma de cryptographie RSA :.

1) L'exposant public e :

Il est propre à une application et est fourni par cette application. De ce fait, il est commun à tous les utilisateurs de cette même application.

2) Les paramètres p et q:

Ils sont générés à l'issu d'un calcul coûteux en temps. Ils ont en général la même longueur (même taille). Cette longueur est classiquement de 512 bits. Pour augmenter la sécurité, cette longueur peut aller de 512 bits à 2048, 2048 bits étant envisagés pour le futur.

3) N est le module public et est calculé à partir de la relation suivante :

$N = p*q$

La clé de l'algorithme est dite de longueur ℓ, lorsque le module public N est de longueur ℓ. Cette longueur est fixée par l'application (ou fournisseur de service).

4) les paramètres e et N forment la clé publique.

5) la clé privée d est calculée à partir de la relation suivante :

$d = 1/e[\mathrm{mod}(p-1)(q-1)]$; $(1/e = e^{-1})$

soit encore $ed \equiv 1$ [(mod ppcm(p-1, q-1)] ; ppcm signifie le plus petit commun multiple,

les paramètres secrets sont formés par le triplet (d, p, q).

6) la forme « normale » de la clé privée est:

(d,N).

6) la forme CRT (Chinese Remainder Theorem) de la clé privée est:

dans ce cas la clé privée comporte 5 paramètres :

p, q

d_p avec $d_p = d \bmod (p-1)$

d_q avec $d_q = d \bmod (q-1)$

I_q avec $I_q = q^{-1} \bmod p$.

Le principe de la génération d'une clé selon le, schéma RSA consiste donc comme on peut le voir, à générer une clé privée d à partir d'un exposant public e (ou clé publique) fixé par l'application, les paramètres p, q étant générés de sorte que p*q = N, la longueur ℓ de N étant fixée.

Lorsque plusieurs applications sont prévues, chaque fournisseur de service fournit son exposant public e et la longueur du module public N, de manière à ce que puisse être générée la clé privée d correspondante.

Ainsi, la mise en œuvre d'un calcul de clé RSA nécessite la connaissance de l'exposant public e et celle de la longueur ℓ de la clé de l'algorithme c'est à dire la longueur du modulo N. Avec les données d'entrée e et ℓ, il reste à générer le couple de nombre premier p et q de manière à ce que ces derniers répondent aux conditions suivantes :

(i) p-1 et q-1 premiers avec e et,

(ii) N= p*q un nombre entier de longueur ℓ.

Ces contraintes sont coûteuses en temps de calcul.

On rappelle à ce propos que la génération et le stockage des clés pour des objets portables tels que les cartes à puce s'effectuent à ce jour des deux manières suivantes :

Selon une première manière, le calcul d'une clé RSA est effectué sur un serveur pour profiter d'une puissance de calcul importante. On requiert alors pour plus de sécurité, un certificat que l'on télécharge avec la clé au sein de l'objet sécurisé lors de sa phase de personnalisation.

Cette solution présente deux inconvénients. :

- d'une part malgré le cadre relativement sécurisé de la personnalisation, il peut y avoir vol ou duplication de la clé du fait de son transfert du serveur vers l'objet sécurisé, et

- d'autre part, chaque clé est chargée dans l'objet dans une phase initiale de personnalisation, ce qui nécessite de prévoir un maximum de clés dans chaque objet pour pouvoir anticiper les futurs besoins.

Dans la pratique, on stocke dans l'objet portable des ensembles de clés et de certificats correspondant à chaque application susceptible d'être utilisée, sans savoir si ces clés seront réellement utiles ultérieurement. Un emplacement mémoire important est utilisé inutilement. Par exemple 0,3 Koctets sont nécessaires pour une clé de RSA de module de 1024bits, alors que les cartes actuelles ont au plus 32Koctets de mémoire programmable. En outre, un nombre important de certificats est acheté à l'entité de confiance ce qui est coûteux.

L'inconvénient ultime mais tout aussi important est qu'il n'est pas possible d'ajouter de nouvelles clés au fur et à mesure que de nouvelles applications pourraient être envisagées.

Selon une deuxième solution, le calcul peut être effectué au sein de l'objet sécurisé. Cela résout le premier inconvénient de la solution précédente mais crée une lourdeur de traitement au niveau de l'objet sécurisé qui possède une faible capacité de calcul.

En effet, lorsque la génération d'une clé RSA est réalisée par un objet portatif tel qu'une carte à puce, si la longueur imposée de clé RSA est de 2048 bits, le calcul prend alors 30 secondes avec un algorithme performant.

Même si ce temps de calcul est acceptable pour certaines applications car on génère les clés RSA une seule fois pour une application donnée, ceci n'est pas satisfaisant pour les services de téléphonie mobile (GSM par exemple) car cette opération se renouvelle à chaque changement de carte SIM et qu'un plus grand nombre de clés doit être prévu pour répondre aux besoins de différentes applications.

Du fait d'un besoin en ressources de calcul important, les clés sont toujours crées durant la phase de personnalisation à partir des exposants publics e fournis par les différentes entités fournisseur de service. Cette étape de calcul ne peut pas être mise en œuvre ultérieurement car elle paralyserait le fonctionnement de l'objet.

De façon pratique ce calcul n'est pas mis en œuvre par la carte. En effet, ce calcul est long et il pourrait ralentir la phase de personnalisation, de plus sa durée est variable et elle pourrait se révéler incompatible avec les procédés de personnalisation des cartes à puce.

D'autre part, cette solution présente toujours le second inconvénient de la solution précédente à savoir la nécessité de ressource mémoire.

La présente invention a pour but de résoudre ces problèmes.

Plus précisément l'invention a pour objectif de résoudre le problème de lourdeur du calcul lié à la gestion de génération de clés ainsi que le problème de manque de flexibilité dû au stockage initial et définitif d'un nombre important de clés et de certificats en phase de personnalisation.

A cette fin, un objet de la présente invention concerne un procédé de génération de clés électroniques d pour procédé de cryptographie à clé publique au moyen d'un dispositif électronique, principalement caractérisé en ce qu'il comprend deux étapes de calcul dissociées :

Etape A

1) Calcul de couples de nombres premiers (p,q) ou de valeurs représentatives de couples de nombres premiers, ce calcul étant indépendant de la connaissance du couple (e,l) dans lequel e est l'exposant public et l la longueur de la clé du procédé de cryptographie, l étant également la longueur du module N dudit procédé,
2) Stockage des couples ou des valeurs ainsi obtenus ;

Etape B

Calcul de la clé d à partir des résultats de l'étape A et de la connaissance du couple (e,l).

Selon une première variante, l'étape A-1) consiste à calculer des couples de nombres premiers (p,q) sans connaissance de l'exposant public e ni de la longueur l de la clé, en utilisant un paramètre Π qui est le produit de petits nombres premiers. De cette manière couple (p,q) obtenu à l'étape A, a une probabilité maximale de pouvoir correspondre à un futur couple (e,l) et permettra de calculer une clé d lors de la mise en œuvre de l'étape B.

Selon une autre variante dépendante de la variante précédente, le calcul A-1) tient compte en plus du fait que e a une forte probabilité de faire partie de l'ensemble $\{3, 17, \ldots, 2^{16+1}\}$, on utilise pour cela dans le

calcul de l'étape A, une graine σ qui permet de calculer non pas des couples (p,q) mais une valeur représentative appelée image des couples (p,q).

Le stockage A-2) consiste alors à mémoriser cette image. Ceci permet de gagner de la place mémoire puisqu'une image est plus petite qu'un nombre premier p ou q par exemple 32 octets comparés à 128 octets.

Selon une troisième variante on effectue un calcul de couples (p,q) pour différents couples (e,l) probables. De façon pratique le paramètre Π va contenir les valeurs usuelles de e par exemple 3, 17.

Selon une quatrième variante l'étape A-1) comprend une opération de compression des couples (p,q) calculés et l'étape A-2) consiste alors à stocker les valeurs compressées ainsi obtenues.

L'étape B comprend la vérification des conditions suivantes pour un couple (e, ℓ) donné:

(i) p-1 et q-1 premiers avec e et,

(ii) N= p*q un nombre entier de longueur ℓ.

Selon un mode de réalisation préféré, l'étape A-1) comprend la génération d'un nombre premier q, le choix d'une limite inférieure B_0 pour la longueur ℓ_0 de ce nombre premier à générer telle que $\ell_0 \geq B_0$ par exemple B_0= 256 bits, et elle comprend en outre les sous-étapes suivantes :

1) -calculer des paramètres v et w à partir des relations suivantes et les mémoriser:

$$v = \sqrt{2^{2\ell_0 - 1}} / \Pi$$

$$w = 2^{\ell_0} / \Pi$$

dans lesquelles Π est mémorisé et correspond au produit des f plus petits nombres premiers, f étant choisi de manière telle que $\Pi \leq 2^{B_0}$,

2)-choisir un nombre j dans l'intervalle des nombres entiers {v,….,w-1} et calculer $\ell = j\ \Pi$;

3)-choisir et enregistrer un nombre premier k de longueur courte par rapport à la longueur d'une clé RSA dans l'intervalle des nombres entiers {0,…., Π-1} , (k, Π) étant co-premiers, ;

4)-calculer $q = k + \ell$,

5)-vérifier que q est un nombre premier, si q n'est pas un nombre premier alors :

a) prendre une nouvelle valeur pour k au moyen de la relation suivante :

$k = a\,k \pmod{\Pi}$; a appartenant au groupe multiplicatif Z^*_{Π} des nombres entiers modulo Π;

b) réitérer à partir de la sous-étape 4).

Avantageusement l'étape B comprend, pour un couple (p,q) obtenu à l'étape A,et un couple (e,l) donné :

- La vérification des conditions suivantes :
  (i) p-1 et q-1 premiers avec e et,
  (ii) N= p*q un nombre entier de longueur ℓ,
- Si le couple (p,q) ne répond pas à ces conditions :
- Choix d'un autre couple et réitération de la vérification jusqu'à ce qu'un couple convienne,
- Calcul de la clé d à partir du couple (p,q) obtenu à l'issue de cette vérification.

L'invention a également pour objet, un objet sécurisé portatif apte à générer des clés électroniques d d'un algorithme de cryptographie de type RSA, caractérisé en ce qu'il comprend au moins :

- Des moyens de communication pour recevoir au moins un couple (e,l),

- Une mémoire pour stoker les résultats d'une étape A consistant à :

Calculer des couples de nombres premiers (p,q) ou de valeurs représentatives de couples de nombres premiers, ce calcul étant indépendant de la connaissance du couple (e,l) dans lequel e est l'exposant public et l la longueur de la clé du procédé de cryptographie, l étant également la longueur du module N dudit procédé,

- Un programme pour mettre en œuvre une étape B consistant à :

Calculer d'une clé d à partir des résultats de l'étape A et de la connaissance d'un couple (e,l),

L'objet sécurisé portatif comprend en outre un programme pour la mise en œuvre de l'étape A, les étapes A et B étant dissociées dans le temps.

L'objet sécurisé portatif pourra être constitué par une carte à puce.

D'autres particularités et avantages de l'invention apparaîtront clairement à la lecture de la description qui est donnée ci-après à titre d'exemple non limitatif et en regard de la figure unique représentant un schéma d'un système de mise en œuvre du procédé.

La suite de la description est faite dans le cadre de l'application de l'invention à un objet portatif de type carte à puce et pour simplifier l'expression on parlera de carte à puce.

Selon le procédé proposé la génération de clés se fait en deux étapes dissociées.

La première Etape A comporte un calcul de couples de nombres premiers (p,q) ou de valeurs représentatives de couples de nombres premiers appelée image.

Les couples (p,q) obtenus sont stockés.

Ce calcul est lourd et il est d'autant plus lourd si on utilise un algorithme de génération de nombres premiers classique.

Il est proposé ici que ce calcul soit effectué de manière indépendante de la connaissance du couple (e,l).

Comme cela va être détaillé dans la suite un mode de réalisation préféré pour mettre en œuvre cette étape permet d'alléger les calculs et de limiter la place mémoire nécessaire pour le stockage des couples (p,q) obtenus en stockant une image de ces couples.

La deuxième Etape B comporte le calcul à proprement parler de la clé d à partir des résultats de l'étape A et de la connaissance du couple (e,l).

Ce calcul comprend, pour un couple (p,q) obtenu à l'étape A, et un couple (e,l) donné :

- La vérification des conditions suivantes :
  (i) p-1 et q-1 premiers avec e et,
  (ii) N= p*q , ce nombre doit être un nombre entier et de longueur ℓ,
- Si un couple (p,q) ne répond pas à ces conditions, on choisit un autre couple et on réitère de la vérification jusqu'à ce qu'un couple convienne parmi les couples obtenus lors de l'étape A.
- On peut procéder alors au calcul de la clé d à partir du couple (p,q) obtenu à l'issue de cette vérification.

La première étape qui correspond à un calcul relativement lourd par rapport à la deuxième étape, peut être exécutée par un autre organe que la carte à

puce par exemple par un serveur. Dans ce cas, les résultats du calcul de cette première étape pourront être chargés sur une carte à puce au moment de la personnalisation.

Le calcul de l'étape A peut également être fait par la carte elle-même à un instant quelconque qui ne gêne pas l'utilisateur de cette carte. Par exemple, ce calcul peut être fait lors de la personnalisation de la carte ou plus tard.

De façon pratique, lors de l'utilisation de la carte, pour obtenir un service, si une clé privée est nécessaire, alors la clé publique est fournie par le fournisseur de service (éventuellement à distance si elle n'est pas déjà stockée dans la carte) afin de générer la clé privée. Cette étape de génération (étape B de calcul) est effectuée de manière rapide par la carte.

On voit donc que de nouvelles applications qui nécessitent le calcul d'une clé privée d peuvent être prévues pour une carte.

On voit également qu'il n'y a pas besoin d'associer un certificat aux couples (p,q) car ils ne sont pas associés à une clé privée.

Ainsi, la génération d'une clé privée peut être faite à bord c'est à dire par la carte elle-même avec un gain d'un facteur 10 en temps d'exécution par rapport aux procédés de génération de clés connus à ce jour.

On va décrire dans ce qui suit un mode préféré de réalisation pour la mise en œuvre de l'étape A. Ce mode de réalisation est particulièrement avantageux pour la mise à bord d'une carte à puce car il permet

d'optimiser à la fois la place mémoire mais aussi le temps de calcul.

Tout d'abord, afin de s'assurer que N=p*q est un entier de ℓ-bit, on choisit p appartenant à l'intervalle :

$$\left[\sqrt{2^{2(l-l0)-1}}, 2^{l-l0}-1\right]$$

Et q appartenant à l'intervalle :

$$\left[\sqrt{2^{2l0-1}}, 2^{l0}-1\right]$$

Pour ℓ_0 compris entre 1 et ℓ.

Ainsi min(p)min(q) est compris entre $2^{\ell 0}$-1 et N, et max(p)max(q) est compris entre N et 2^{ℓ} comme cela est demandé.

De cette façon, la condition ii) ci-dessus mentionnée se réduit à rechercher des nombres premiers dans l'intervalle :

$$\left[\sqrt{2^{2l0-1}}, 2^{l0}-1\right]$$

La solution proposée exploite le paramètre Π. Ce paramètre Π est le produit de petits nombres premiers dans lequel on peut trouver notamment 3, 17, 2^{16+1}, nombres premiers généralement utilisés comme exposants publics. Ainsi, la probabilité pour qu'un couple (p,q) corresponde à un futur couple (e,1) donné, déjà très élevée, augmente encore lorsque Π comporte de telles valeurs.

On choisit les f plus petits nombres premiers, f étant choisi de manière telle que $\Pi_i p_i \leq 2B_0$, B_0 est la

borne inférieure choisie pour ℓ_0. par exemple on peut choisir B_0 égal à 256 bits.

Π est égal au produit : 2.3…..191 et est inférieur à 2^{256}.

On peut alors mémoriser cette valeur Π dans la carte par exemple comme une constante dans la mémoire morte de programme.

La première phase du procédé consiste à générer et à enregistrer un nombre premier k de longueur courte par rapport à la longueur d'une clé RSA dans l'intervalle des nombre entiers $\{0,\ldots., \Pi-1\}$, (k, Π) étant copremiers, c'est à dire n'ayant pas de facteur commun.

La deuxième phase consiste ensuite à partir de ce nombre k à construire le premier candidat q qui satisfait la condition d'être copremier avec Π .

Si ce premier candidat ne satisfait pas cette condition, alors il est mis à jour c'est à dire qu'un autre candidat est choisi jusqu'à ce qu'une valeur de q satisfaisant à la condition soit trouvée.

On va présenter dans la suite les différentes étapes de l'algorithme de génération d'un nombre premier entrant dans le calcul d'une clé RSA selon l'invention.

L'algorithme proposé fonctionne quelle que soit la longueur lo donnée pour le nombre premier q qui doit être généré.

La génération du nombre premier p est identique, il suffit de remplacer q par p dans les étapes qui vont être développées et de remplacer lo par l-lo.

Après avoir fixé la limite Bo, on calcule les nombres premiers uniques v et w satisfaisant les conditions suivantes:

$$\sqrt{2}\,2^{\ell_0-1} \le v\Pi \le \sqrt{2}\,2^{\ell_0-1} + \Pi,$$
$$2^{\ell_0} - \Pi \le w\Pi \le 2^{\ell_0}$$

Ceci, se traduit par le calcul de v et w par les relations suivantes :

$$v = \sqrt{2}\,2^{\ell_0-1} / \Pi$$
$$w = 2^{\ell_0} / \Pi$$

Puis après avoir pris k appartenant au groupe multiplicatif $Z^*\Pi$ des nombres entiers modulo Π, on construit le premier candidat q tel que,

$q = k + j\,\Pi$ pour tout j appartenant à l'intervalle [v, w-1] .

Comme justement k apparient à $Z^*\Pi$, la probabilité pour avoir un premier candidat q premier, est élevée. Si ce n'est pas le cas, on met à jour k en prenant k égal à $ak \pmod{\Pi}$, a appartenant au groupe $Z^*\Pi$ et on réitère jusqu'à trouver une valeur de q correspondant à un nombre premier.

Une manière de tester la primalité d'un nombre est par exemple d'utiliser le test de Rabin-Miller.

Les différentes étapes de l'algorithme proposé sont précisément les suivantes :

1) -calculer des paramètres v et w à partir des relations suivantes et les mémoriser:

$$v = \sqrt{2}\,2^{\ell_0-1} / \Pi$$
$$w = 2^{\ell_0} / \Pi$$

dans lesquelles Π est mémorisé et correspond au produit des f plus petits nombres premiers, f étant choisi de manière telle que $\Pi \le 2^{B_0}$,

2)-choisir un nombre j dans l'intervalle des nombres entiers $\{v, \ldots, w-1\}$ et calculer $\ell = j\,\Pi$;

3)-choisir et enregistrer un nombre premier k de longueur courte par rapport à la longueur d'une clé RSA dans l'intervalle des nombres entiers $\{0,\ldots., \Pi-1\}$, (k, Π) étant co-premiers, ;

4)-calculer $q= k+\ell$,

5)-vérifier que q est un nombre premier, si q n'est pas un nombre premier alors :

a) prendre une nouvelle valeur pour k au moyen de la relation suivante :

$k = a\,k \pmod{\Pi}$; $\underline{a}$ appartenant au groupe multiplicatif Z^*_Π des nombres entiers modulo Π;

b) réitérer à partir de l'étape 4) ;

6) enregistrer a, k' j pour les utiliser afin de retrouver q et ensuite exploiter q pour l'utiliser lors d'un calcul ultérieur de génération d'une clé RSA.

Au lieu de stocker la valeur de q on va procéder avantageusement comme décrit dans la suite.

Une manière simple de mettre en œuvre cet algorithme peut consister pour chaque longueur de clé RSA envisagée, de stocker les valeurs de k et j de manière à re construire q.

Plutôt que de choisir un nombre aléatoire j comme indiqué à l'étape 2) un autre mode de réalisation peut consister à construire j à partir d'un nombre aléatoire court.

On prend par exemple un nombre de longueur 64-bit, que l'on désigne par graine et que l'on dénote σ. Cette graine est alors prise comme valeur d'entrée d'un générateur de nombres pseudo-aléatoires PRNG, lequel va permettre de générer j.

j est alors défini comme $PRNG_1(\sigma) \pmod{(w-v)+v}$.

Ce mode d'exécution permet de réduire considérablement les besoins en place mémoire car il n'y a à stocker que les valeurs de σ et de k en mémoire

EEPROM. La valeur de Π est en mémoire morte (dans le programme de calcul).

On peut encore réduire les besoins en place mémoire en constatant que : si $k_{(o)}$ est la première valeur de k appartenant au groupe $Z^*\prod$, alors, les nombres premiers générés ont la forme :

$q = a^{f-1} k_{(o)} \bmod \Pi + j\,\Pi$

f étant le nombre d'échec du test de l'étape 4).

Cette valeur $k_{(o)}$ qui appartient au groupe $Z^*\prod$, peut être facilement calculée à partir d'une graine aléatoire courte comme σ par exemple et en utilisant la fonction de Carmichael de Π^2 dénotée $\lambda(\Pi)$.

En utilisant cette fonction on peut exprimer $k_{(o)}$ par la relation suivante :

$k_{(o)} = [PRNG_2(\sigma) + b^{PRNG3(\sigma)} (PRNG_2(\sigma)^{\lambda(\Pi)} - 1)] \pmod{\prod}$

b étant un élément d'ordre $\lambda(\Pi)$ appartenant à $Z^*\prod$.

Ces deux modes d'exécution permettent de réduire les besoins en place mémoire puisqu'on ne va devoir stoker dans ce cas, que la valeur de la graine σ et différentes valeurs de f pour les longueurs désirées de clés.

Pour des clés RSA de modulo supérieur à 2048 bits, les expériences numériques qui ont été faites par les inventeurs montrent que f est égal à 2^8. Ceci signifie que f peut être codé sur 1 byte soit 8 octets.

A titre d'exemple, pour générer des clés RSA de longueur allant de 512 à 2048 bits avec une granulité de 32 bits, il y a 49 longueurs de clé possibles. Il est donc nécessaire de stocker sur la carte un byte soit 8 octets correspondant à la valeur de σ. Il est également nécessaire de stocker les valeurs de f pour les nombres premiers p et q soit 2*49=98 octets. Ceci

fait au total 106 bytes soit 848 bits en mémoire EEPROM.

Un dernier mode d'exécution permettant de réduire la place mémoire, consiste à stoker dans le programme de calcul, c'est à dire en mémoire de programme, plusieurs valeurs de Π et les valeurs de $\lambda(\Pi)$ correspondantes pour différentes longueurs de clés envisagées. On peut remarquer qu'une grande valeur de Π conduit aux plus petites valeurs pour f.

Le nombre premier q généré selon l'étape 4) par l'algorithme qui vient d'être décrit satisfait comme on l'a vu précédemment à la condition :

$q= a^{f-1} k_{(o)} \bmod \Pi + j*\Pi$

Si e divise Π on peut exprimer q par la relation suivante :

$q= a^{f-1} k_{(o)} \bmod (e)$

Afin que la condition i) énoncée au début de la description soit remplie, il faut choisir a tel que $a=1 \pmod e$ et forcer $k_{(o)}$ de manière à ce qu'il soit différent de $1 \pmod e$.

Ainsi le nombre premier q obtenu satisfait la relation $q= k_{(o)}$ différent de $1 \pmod e$.

La génération du nombre premier p est identique, q est remplacé par p dans les étapes qui ont été développées et lo par 1-lo.

Comme cela a été dit, le programme mettant en œuvre le procédé de la carte n'a pas besoin de connaître à priori l'exposant public e. Cet exposant peut donc être fourni à tout moment par une application chargée dans la carte.

Toutefois, on sait que pour la plupart des applications (plus de 95%), les valeurs de e utilisées sont les valeurs $\{3, 17, 2^{16}+1\}$.

Afin de couvrir le plus grand nombre d'applications, on va de façon préférentielle choisir a tel que $a= 1 \bmod(\{3, 17, 2^{16}+1\})$ et forcer $k_{(o)}$ différent de cette valeur : $1 \bmod(\{3, 17, 2^{16}+1\})$.

On choisit par exemple comme candidat possible pour a, le nombre premier $R= 2^{64}-2^{32}+1$ à condition que le plus grand commun diviseur de Π et de R soit égal à 1.

La condition requise pour $k_{(o)}$ peut être obtenue par le théorème du reste chinois.

Comme cela a été dit une autre alternative peut consister pour l'étape A-1) à calculer des couples de nombres premiers (p,q) pour différents couples (e,1) probables.

En conclusion, l'invention propose un procédé en deux étapes dissociées, la deuxième étape très rapide par rapport aux solutions connues, peut être exécutée en temps réel. Ce procédé est également peu coûteux en place mémoire.

En outre, il n'y a pas de limite pour de nouvelles applications non prévues à la personnalisation de la carte.

REVENDICATIONS

1. Procédé de génération de clés électroniques d pour procédé de cryptographie à clé publique au moyen d'un dispositif électronique, principalement caractérisé en ce qu'il comprend deux étapes de calcul dissociées :

Etape A

1) Calcul de couples de nombres premiers (p,q) ou de valeurs représentatives de couples de nombres premiers, ce calcul étant indépendant de la connaissance du couple (e,l) dans lequel e est l'exposant public et l la longueur de la clé du procédé de cryptographie, l étant également la longueur du module N dudit procédé,
2) Stockage des couples ou des valeurs ainsi obtenus ;

Etape B

Calcul d'une clé d à partir des résultats de l'étape A et de la connaissance du couple (e,l).

2. Procédé de génération de clés électroniques selon la revendication 1, caractérisé en ce que l'étape A-1) consiste à calculer des couples de nombres premiers (p,q) sans connaissance de l'exposant public e ni de la longueur l de la clé, en utilisant un paramètre Π qui est le produit de petits nombres premiers, de manière à ce que chaque couple (p,q) ait une probabilité maximale de pouvoir correspondre à un futur couple (e,l) et puisse permettre de calculer une clé d.

3. Procédé de génération de clés électroniques selon la revendication 2, caractérisé en ce que le

calcul de l'étape A-1) tient compte en plus du fait que e a une forte probabilité de faire partie de l'ensemble $\{3, 17, \ldots, 2^{16+1}\}$, on utilise pour cela dans ce calcul une graine σ qui permet de calculer non pas des couples (p,q) mais une valeur représentative appelée image des couples (p,q).

4. Procédé de génération de clés électroniques selon la revendication 1 et 3, caractérisé en ce le stockage A-2) consiste à mémoriser l'image des couples.

5. Procédé de génération de clés électroniques selon la revendication 1, caractérisé en ce que l'étape A-1) consiste à calculer des couples de nombres premiers (p,q) pour différents couples (e,l) probables.

6. Procédé de génération de clés électroniques selon les revendications 2 et 5, caractérisé en ce que le paramètre Π contient les valeurs usuelles de l'exposant public e par exemple 3, 17.

7. Procédé de génération de clés électroniques selon la revendications 1, caractérisé en ce que l'étape A-1) comprend une opération de compression des couples (p,q) calculés et l'étape A-2) consiste alors à stocker les valeurs compressées ainsi obtenues.

8. Procédé de génération de clés électroniques selon la revendication 1, caractérisé en ce que l'étape A-1) comprend la génération d'un nombre premier q, pour lequel on fixe une limite inférieure B_0 pour la longueur ℓ_0 de ce nombre premier à générer, telle que $\ell_0 \geq B_0$ par exemple B_0= 256 bits, et en ce qu'elle comprend les sous étapes suivantes :

1) -calculer des paramètres v et w à partir des relations suivantes et les mémoriser:

$v = \sqrt{2}^{2\ell_0 - 1} / \Pi$

$w = 2^{\ell_0} / \Pi$

dans lesquelles Π est mémorisé et correspond au produit des f plus petits nombres premiers, f étant choisi de manière telle que $\Pi \leq 2^{B_0}$,

2)-choisir un nombre j dans l'intervalle des nombres entiers $\{v, \ldots, w-1\}$ et calculer $\ell = j\ \Pi$;

3)-choisir et enregistrer un nombre premier k de longueur courte par rapport à la longueur d'une clé RSA dans l'intervalle des nombres entiers $\{0, \ldots, \Pi - 1\}$, (k, Π) étant co-premiers, ;

4)-calculer $q = k + \ell$,

5)-vérifier que q est un nombre premier, si q n'est pas un nombre premier alors :

a) prendre une nouvelle valeur pour k au moyen de la relation suivante :

$k = a\,k \pmod{\Pi}$; a appartenant au groupe multiplicatif Z^*_Π des nombres entiers modulo Π;

b) réitérer à partir de l'étape 4) ;

9. Procédé de génération de clés électroniques selon les revendications 3 et 8, caractérisé en ce que les nombres j et k peuvent être générés à partir de la graine σ stockée en mémoire.

10. Procédé de génération de clés électroniques selon la revendication 8, caractérisé en ce que le nombre premier p est généré en réitérant toutes les sous étapes précédentes en remplaçant q par p et en remplaçant ℓ_0 par $\ell - \ell_0$.

11. Procédé de génération de clés électroniques selon l'une quelconque des revendications précédentes, caractérisé en ce que :

L'étape B comprend, pour un couple (p,q) obtenu à l'étape A, :

- La vérification des conditions suivantes :
  (i) p-1 et q-1 premiers avec e donné et,
  (ii) N= p*q un nombre entier de longueur ℓ donnée,
- Si le couple (p'q) ne répond pas à ces conditions :
- Choix d'un autre couple et réitération de la vérification jusqu'à ce qu'un couple convienne,
- Calcul de la clé d à partir du couple (p,q) obtenu.

12. Objet sécurisé portatif apte à générer des clés électroniques d d'un algorithme de cryptographie de type RSA, caractérisé en ce qu'il comprend au moins :

- Des moyens de communication pour recevoir au moins un couple (e,l),
- Une mémoire pour stoker les résultats d'une étape A consistant à :

  Calculer des couples de nombres premiers (p,q) ou de valeurs représentatives de couples de nombres premiers, ce calcul étant indépendant de la connaissance du couple (e,l) dans lequel e est l'exposant public et l la longueur de la clé du procédé de cryptographie, l étant également la longueur du module N de ce p,
- Un programme pour mettre en œuvre une étape B consistant à :

  Calculer une clé d à partir des résultats de l'étape A et de la connaissance d'un couple (e,l),

13. Objet sécurisé portatif selon la revendication 12, caractérisé en ce qu'il comprend en outre un programme pour la mise en œuvre de l'étape A, les étapes A et B étant dissociées dans le temps.

14. Objet sécurisé portatif selon la revendication 13, caractérisé en ce que le programme de mise en œuvre de l'étape A met en œuvre les sous-étapes :

1) -calculer des paramètres v et w à partir des relations suivantes et les mémoriser:

$$v = \sqrt{2^{2\ell_0 - 1}} / \Pi$$

$$w = 2^{\ell_0} / \Pi$$

dans lesquelles Π est mémorisé et correspond au produit des f plus petits nombres premiers, f étant choisi de manière telle que $\Pi \leq 2^{B_0}$, B_0 est une limite inférieure fixée pour la longueur ℓ_0 du nombre premier à générer telle que $\ell_0 \geq B_0$ par exemple B_0= 256 bits,

2)-choisir un nombre j dans l'intervalle des nombres entiers {v,….,w-1} et calculer $\ell = j\ \Pi$;

3)-choisir et enregistrer un nombre premier k de longueur courte par rapport à la longueur d'une clé RSA dans l'intervalle des nombres entiers {0,…., Π-1} , (k, Π) étant co-premiers, ;

4)-calculer q= k+ℓ,

5)-vérifier que q est un nombre premier, si q n'est pas un nombre premier alors :

a) prendre une nouvelle valeur pour k au moyen de la relation suivante :

$k = a\ k \pmod{\Pi}$; a appartenant au groupe multiplicatif Z^*_Π des nombres entiers modulo Π;

b) réitérer à partir de l'étape 4).

15. Objet sécurisé portatif selon la revendication 12 ou 13 ou 14, caractérisé en ce qu'il est constitué par une carte à puce.

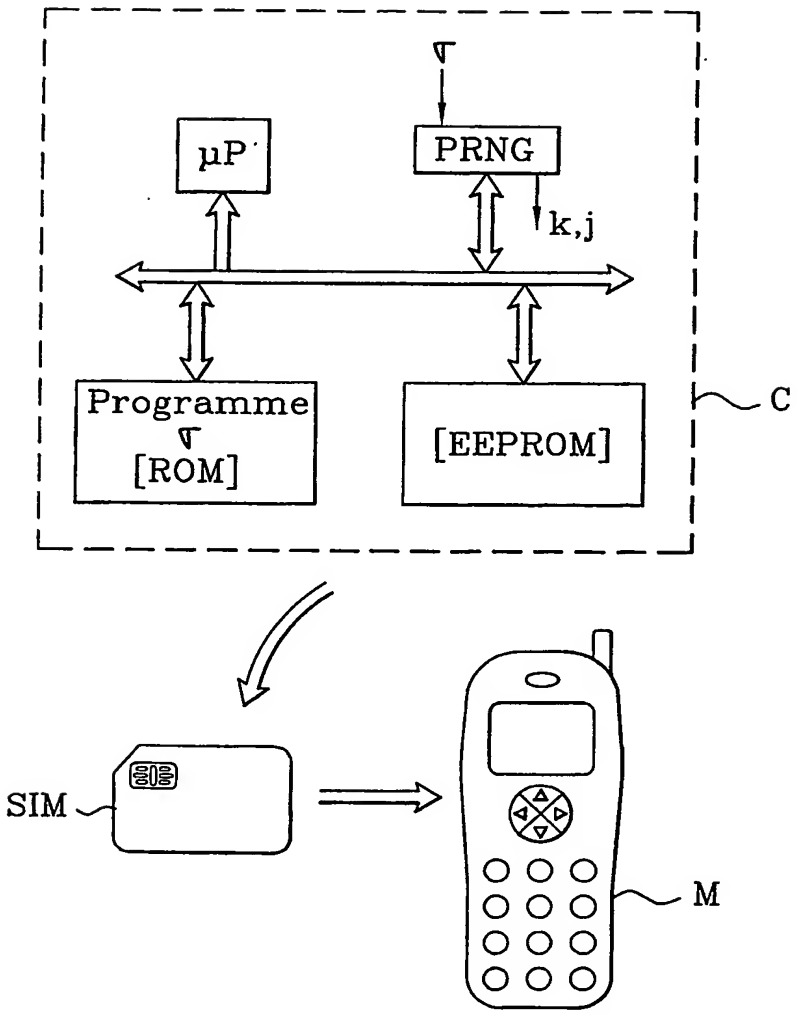

Figure unique